CSU34031 Project 1

Student name: Davy Nolan

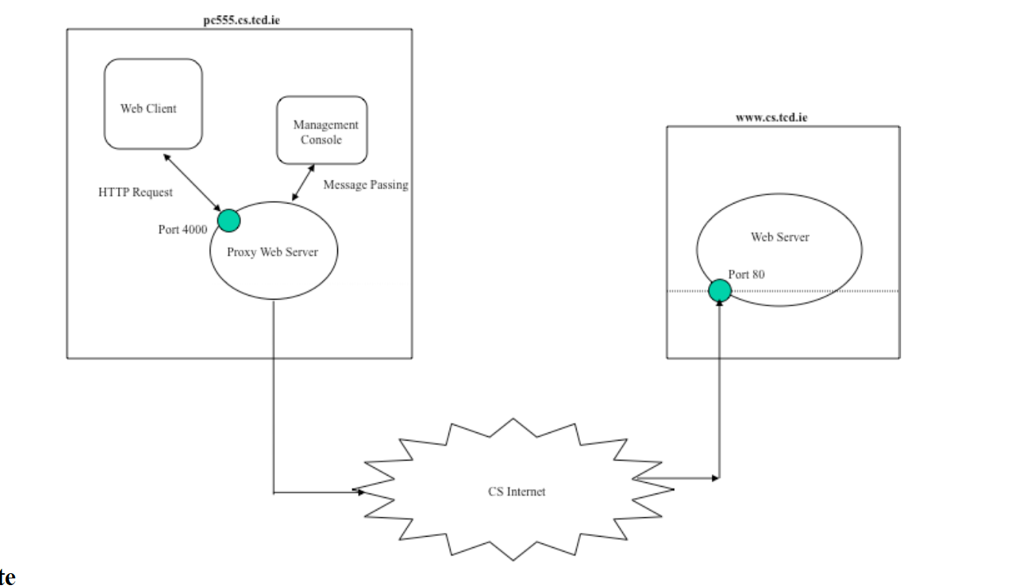
Student number: 17330208

**Introduction**

The objective of this project is to implement a web proxy server using any language of preference. I have decided to use python for my implementation.

**Requirements**

The program should be able to:

1. Respond to HTTP and HTTPS requests, and should display each request on a management console. It should forward the request to the Web server and relay the response to the browser.
2. Handle Websocket connections.
3. Dynamically block selected URLs via the management console.
4. Efficiently cache requests locally and thus save bandwidth. You must gather timing and bandwidth data to prove the efficiency of your proxy.
5. Handle multiple requests simultaneously by implementing a threaded server.

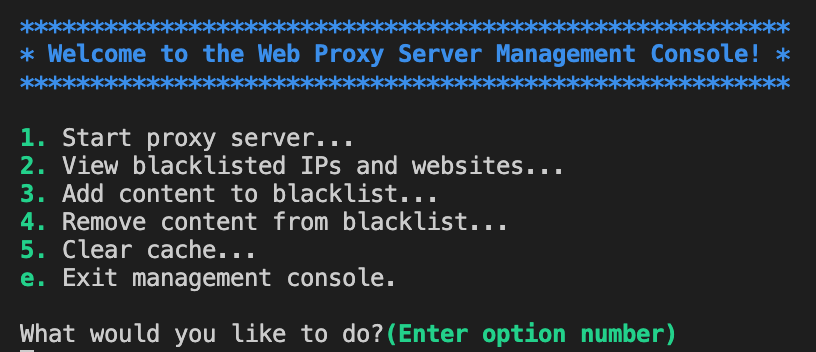


**Implementation**

I decided to split this project into two separate python programs that work together as one:

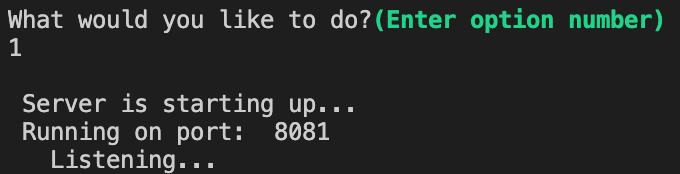
1. **Web\_proxy\_server.py** – This holds all the code involved in processing the HTTP/HTTPS requests, handling cache and also checking to see if a website or IP is blacklisted(blocked).
2. **Mng\_console.py –** This program displays a UI in the console for handling processes in web\_proxy\_server.py in a more user-friendly way.

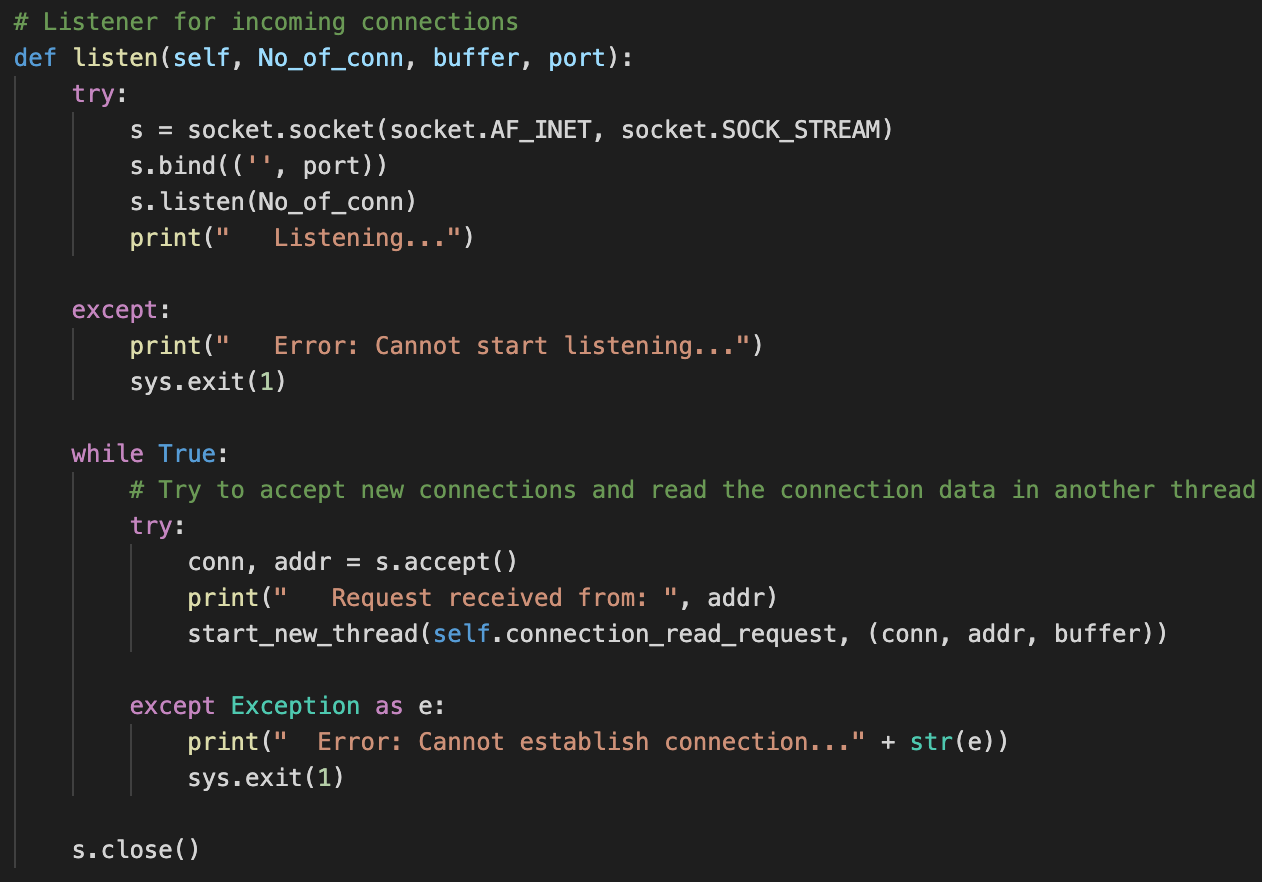
**Code Explanation**

****Running “python3 mng\_console.py” will open up the UI for the program. You will be met by 6 options: Entering in any option from the list will perform the corresponding function explained.

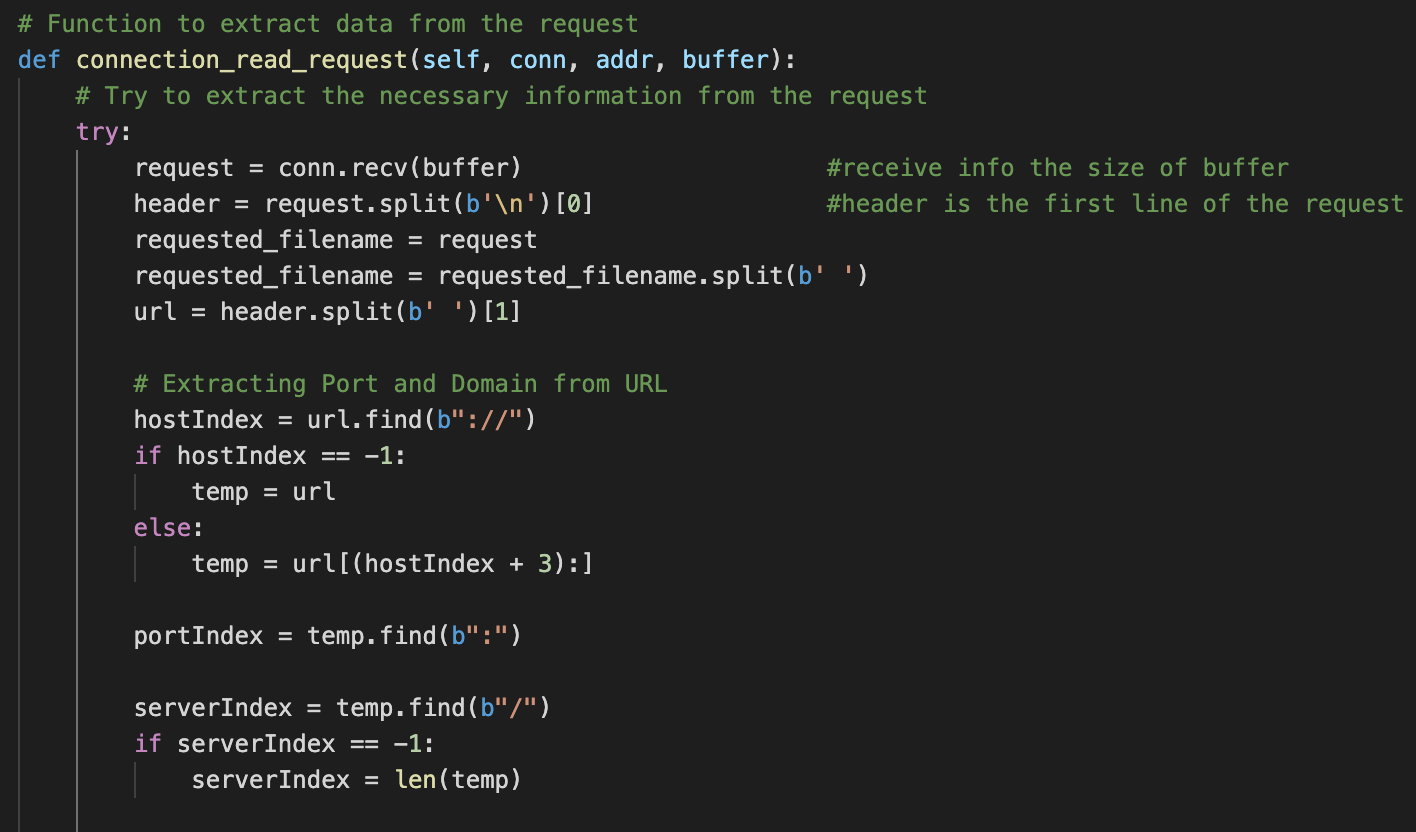
**Running the Server**

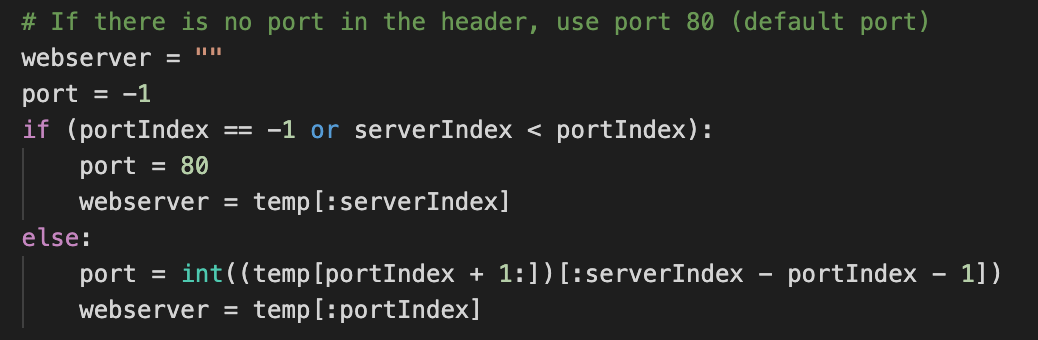
Selecting “option 1” from the management console will begin running the server. The port number is hardcoded into the start\_server() function in web\_proxy\_server.py.

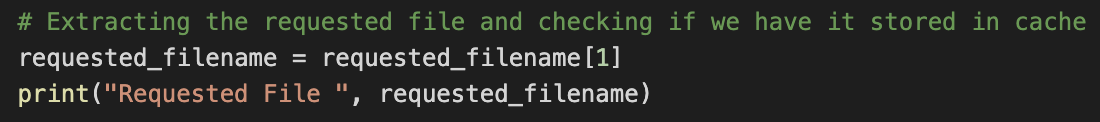


Once running, the server will call the listen() function and begin listening for the client (the websocket). Once a connection is received, it creates a new thread, calling the connection\_read\_request() function.

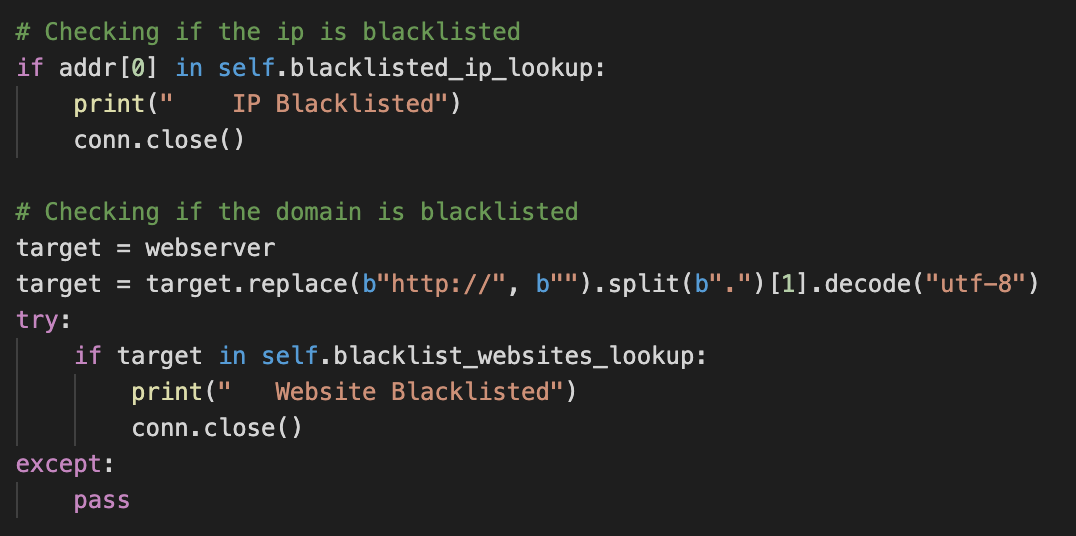
The connection\_read\_request() function extracts data from the request. In the screenshot below it is extracting both the port and domain from the URL.

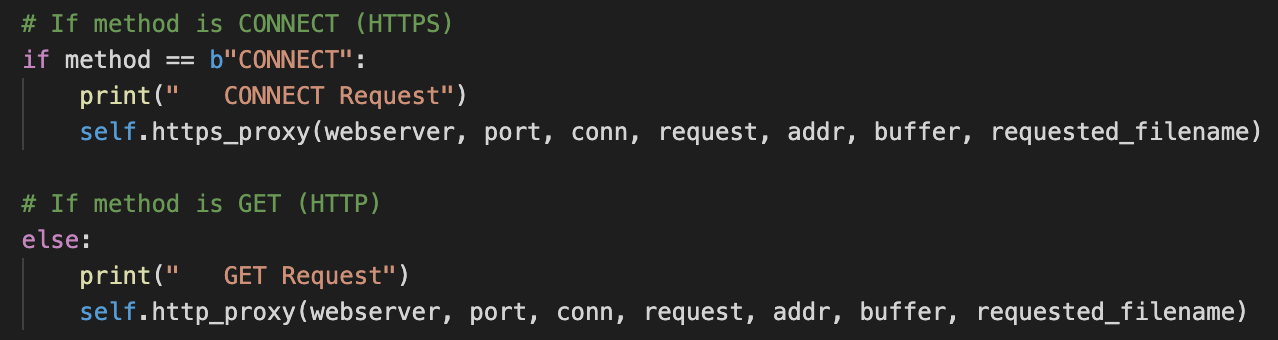


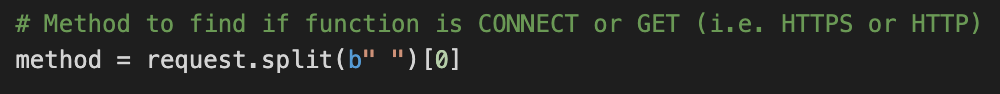
If there’s no port to extract from the header, it uses the default port 80.

****This function also extracts the requested file name to check if it’s stored in cache already.

It also checks to see if the IP and/or the domain is blacklisted and closes the connection if so.



Lastly, the function checks to see if the request is either a CONNECT request or a GET request. CONNECT implies that it is HTTPS and GET implies that it’s HTTP. Once the program checks this, it calls either the https\_proxy() or http\_proxy() ****function depending on the request type.

****

The screenshot below shows how the http\_proxy() function works in handling HTTP requests. It checks if the requested file is stored in cache.

****

If the file is not stored in cache (no cache hit) the server requests it from the web and stores it in cache.

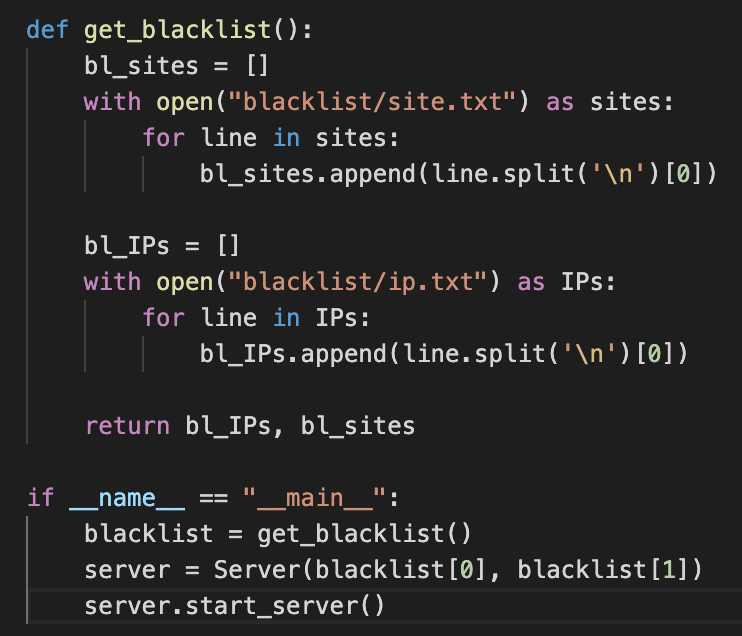
****

Similarly, to the above implementation, HTTPS requests are handled more or less the same way.

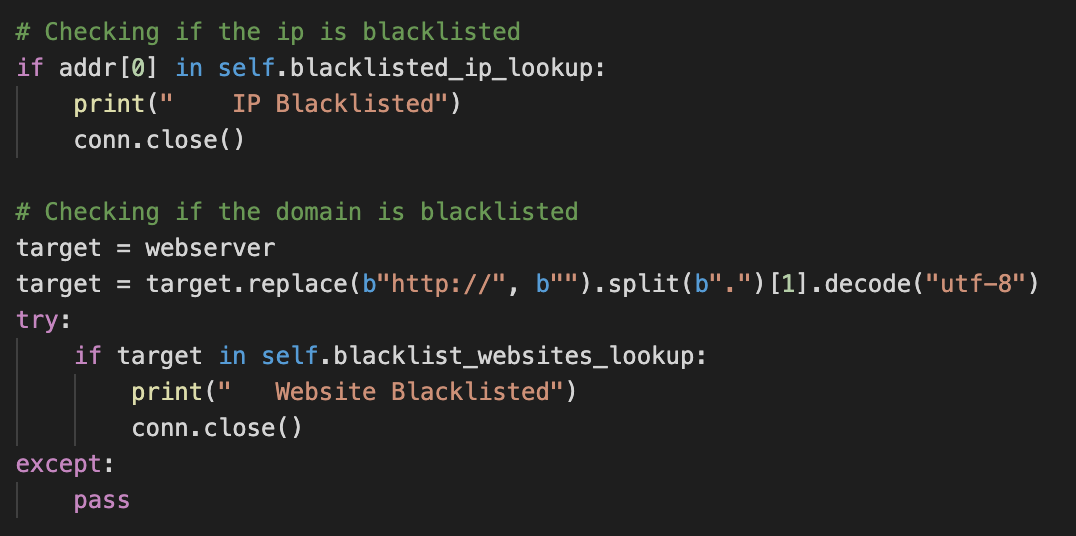
****

**Blacklisting (blocking)**

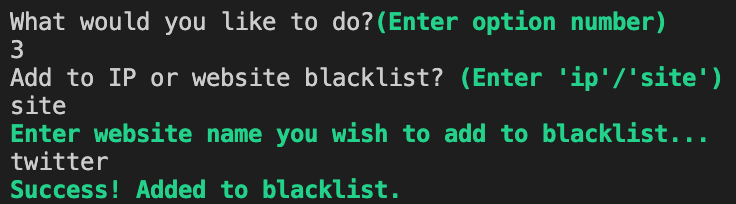
For the blacklisted IPs and websites, there are two separate text files for storing this information. Blacklisted IPs can be listed in “blacklist/ip.txt” and websites can be listed in “blacklist/site.txt”. These text files are read into the server program and are added to lists within the server.

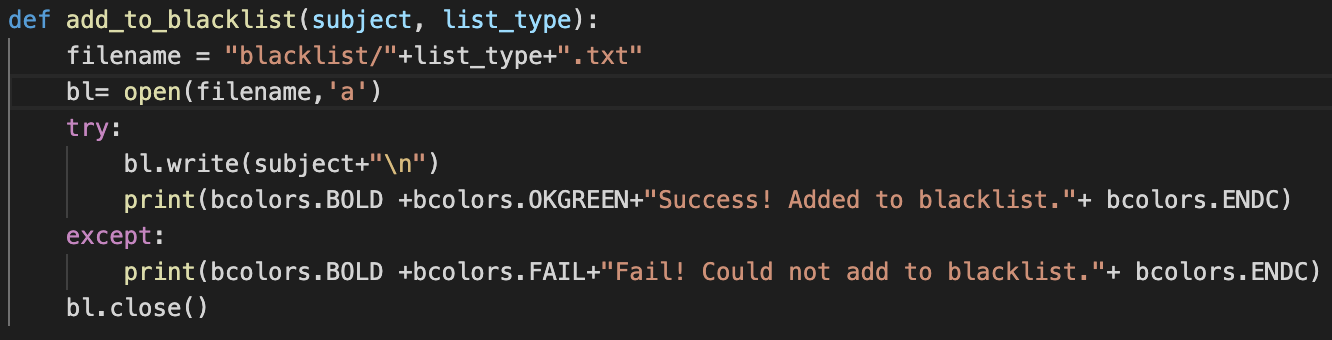


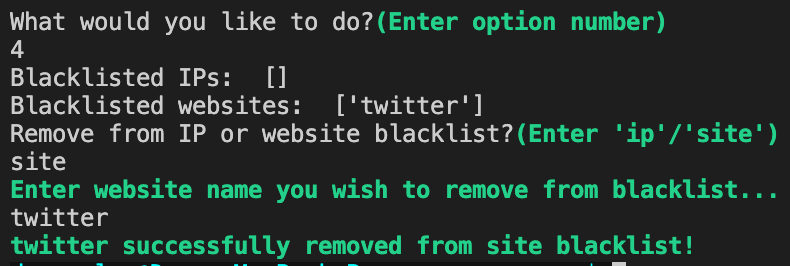
The server will then refer to these lists to check to see if the IP or website being searched is blocked and will act accordingly. As seen in the screenshot below, the program strips the webserver URL to just be left with the name of the website ((i.e.) <https://www.facebook.com> becomes “facebook”). This makes it possible to blacklist HTTPS sites. It then searches for this website name in the website blacklist, and if it is in the blacklist, it closes the connection and prints to the console.

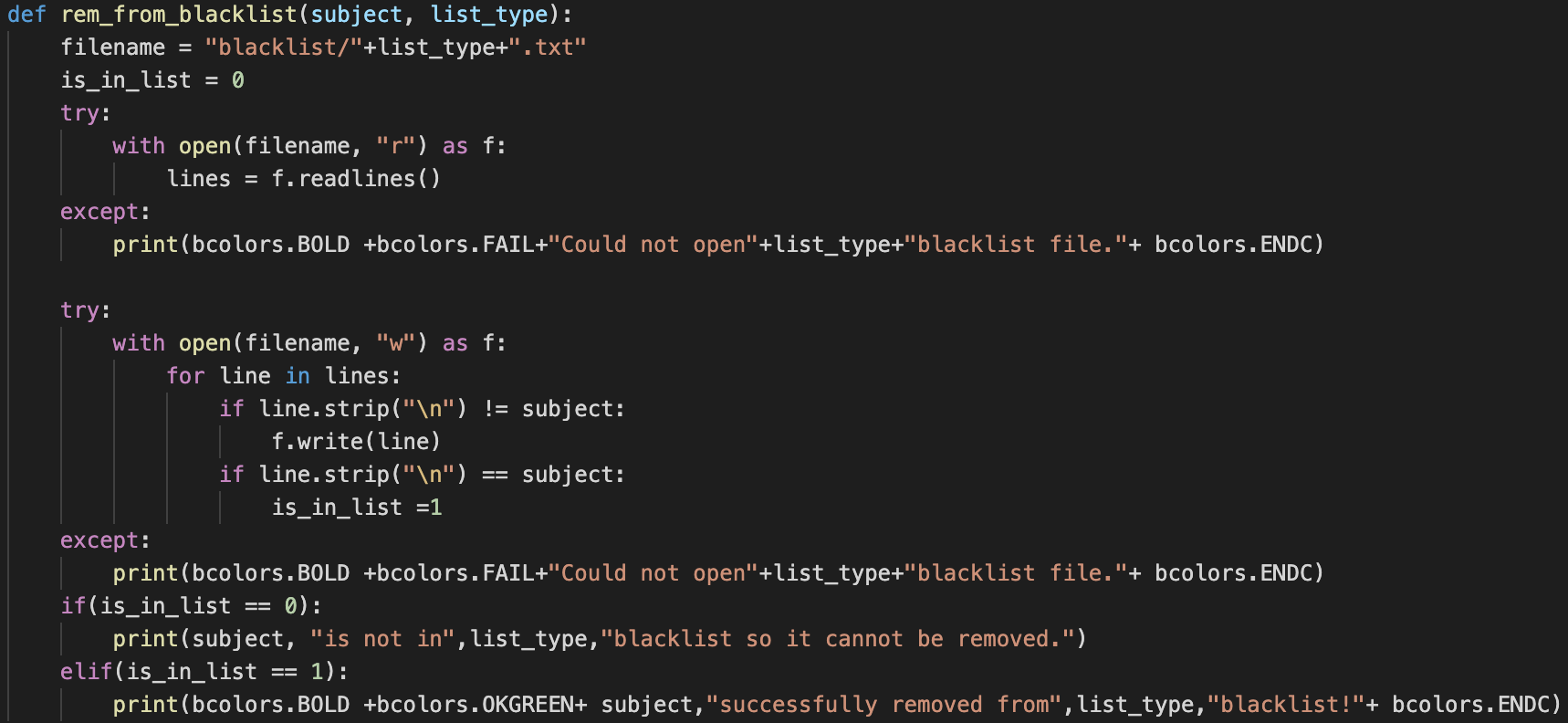


To add new content to the blacklists, you can easily do this via the management console. Selecting “option 3” will prompt you to enter what list you want to add to (either IP or website name) and then ask you to enter either the IP address or website you wish to block. (The example below demonstrates adding a website only but it is the same idea for IP, you just enter ‘ip’ instead of ‘site’)

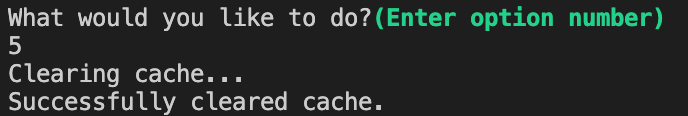
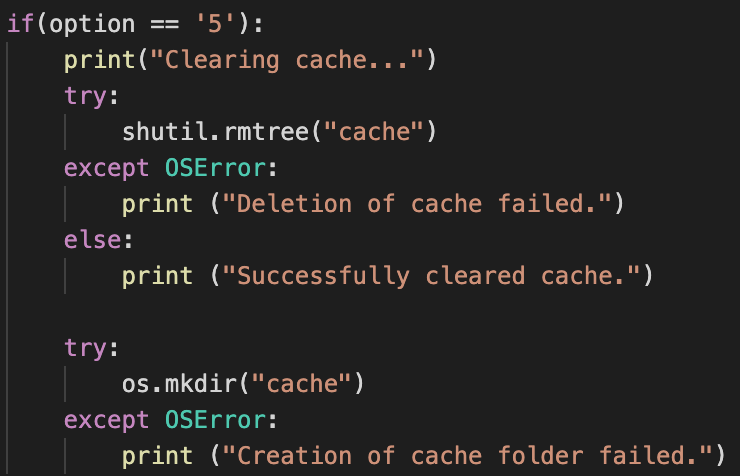




Content can also be removed from the blacklists using the management console by selecting “option 4”. This will display both the IP and website blacklists with their current content. You just have to enter what you want to remove.



**Clear Cache**

Cache can be freed up by selecting “option 5” in the management console. This will practically delete the cache directory and create a new one.